EtherCAT Slave Editor

An editor for creating SOES slaves

rt-labs AB

Background

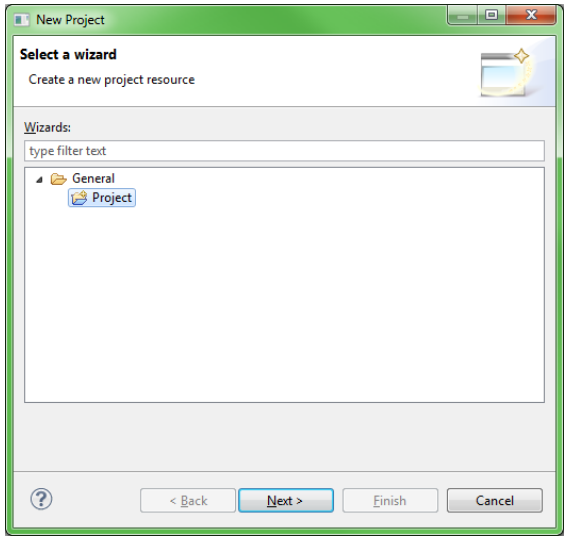
- When developing an EtherCAT slave it is often tiresome and error prone to update the object dictionary, ESI file and EEPROM and keep them aligned
- The EtherCAT Slave Editor is a tool for helping out with this process
- The EtherCAT Slave Editor generates ESI files, EEPROM files, object dictionaries and data structures for slaves using the SOES stack by providing a easy to use interface

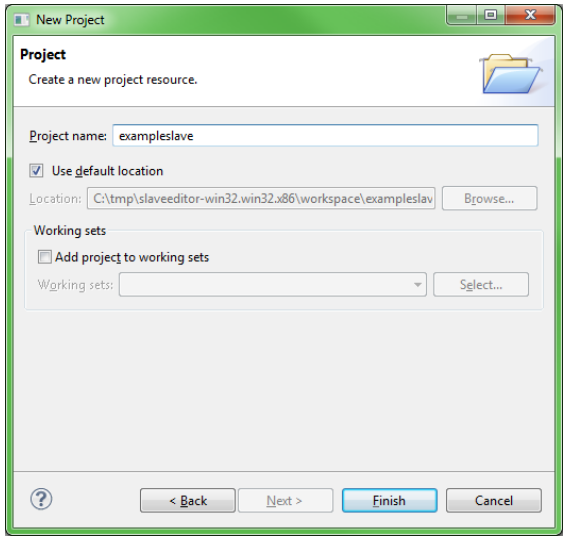
Preparation

- Download the EtherCAT Slave Editor from `http://download.rt-labs.com/ethercat/slaveeditor`
- Unzip and run
- The Slave Editor needs the Java Runtime Environment (JRE): `http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`

Create a project

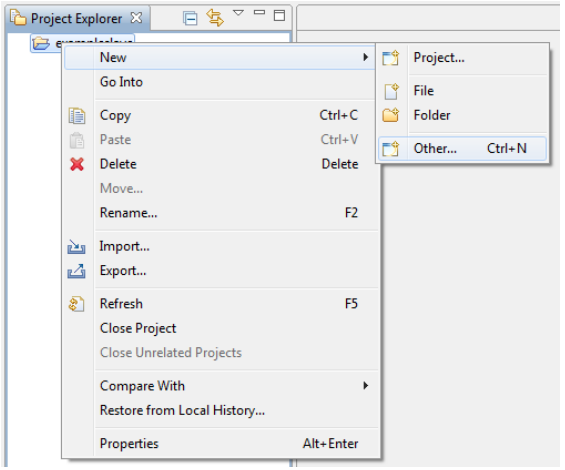
- First we need to create a project
- Right click in the `Project Explorer`
- Choose `New` / `Project`

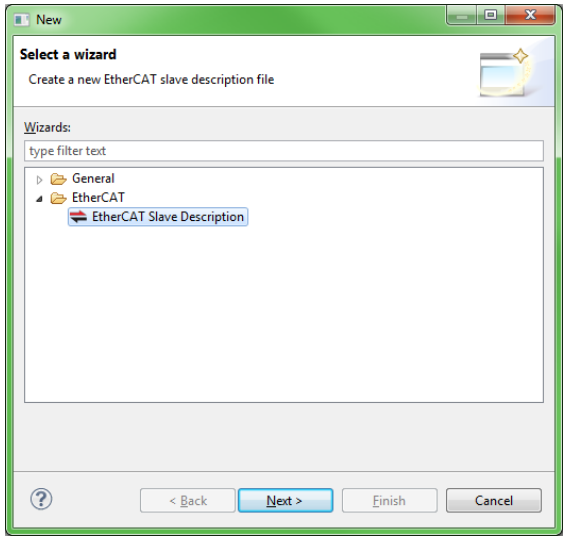




Use the EtherCAT Slave Editor

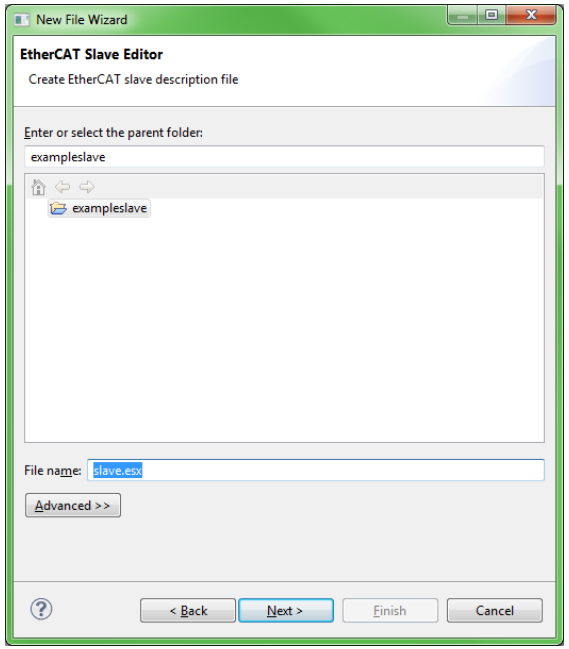
- To work with the EtherCAT Slave Editor we need to add a slave description file

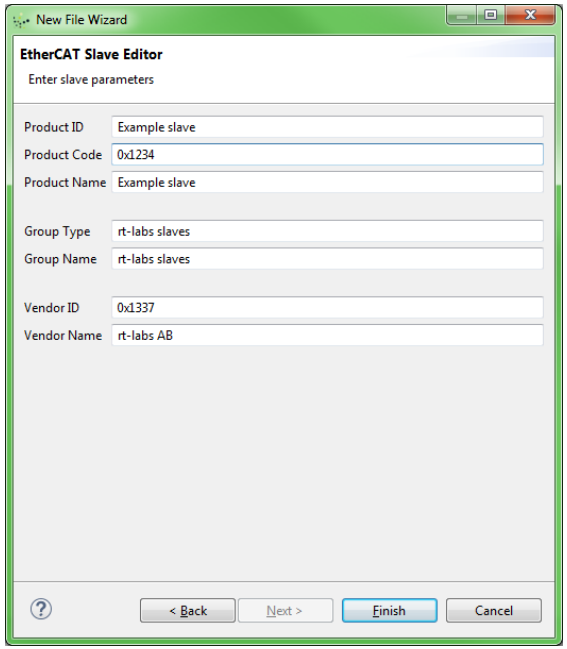




Use the EtherCAT Slave Editor

- Choose a file destination and fill in some basic administrative data
- Parts of the data will be remembered when creating other slaves

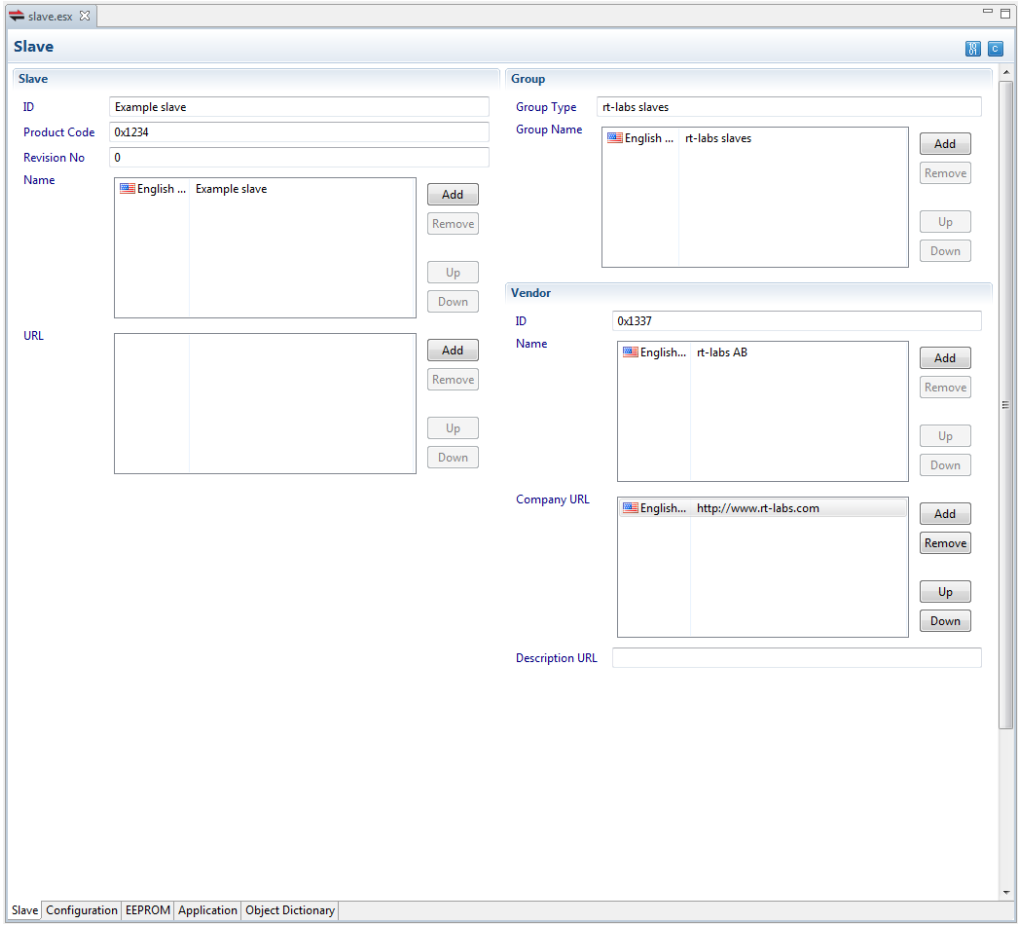




Slave configuration

Administrative data

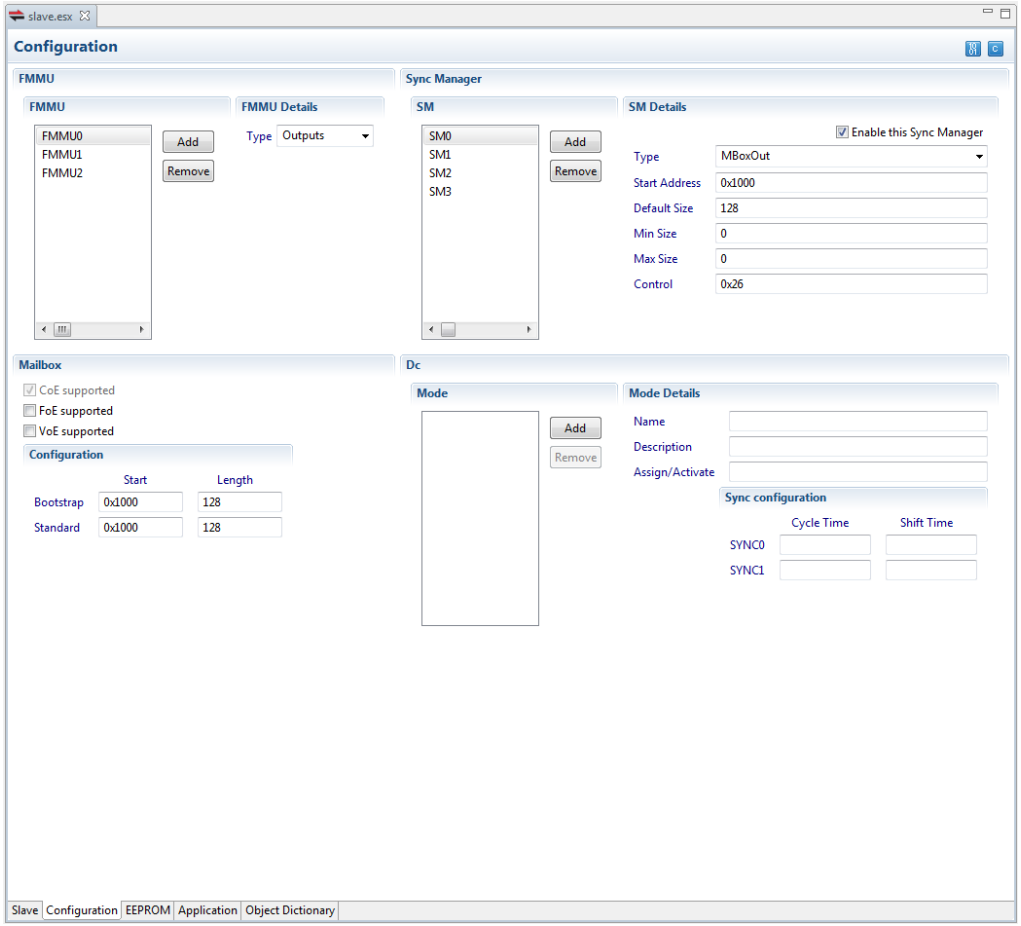
- This tab contains information previously provided in the wizard



Slave configuration

FMMU/SM/MBOX/DC

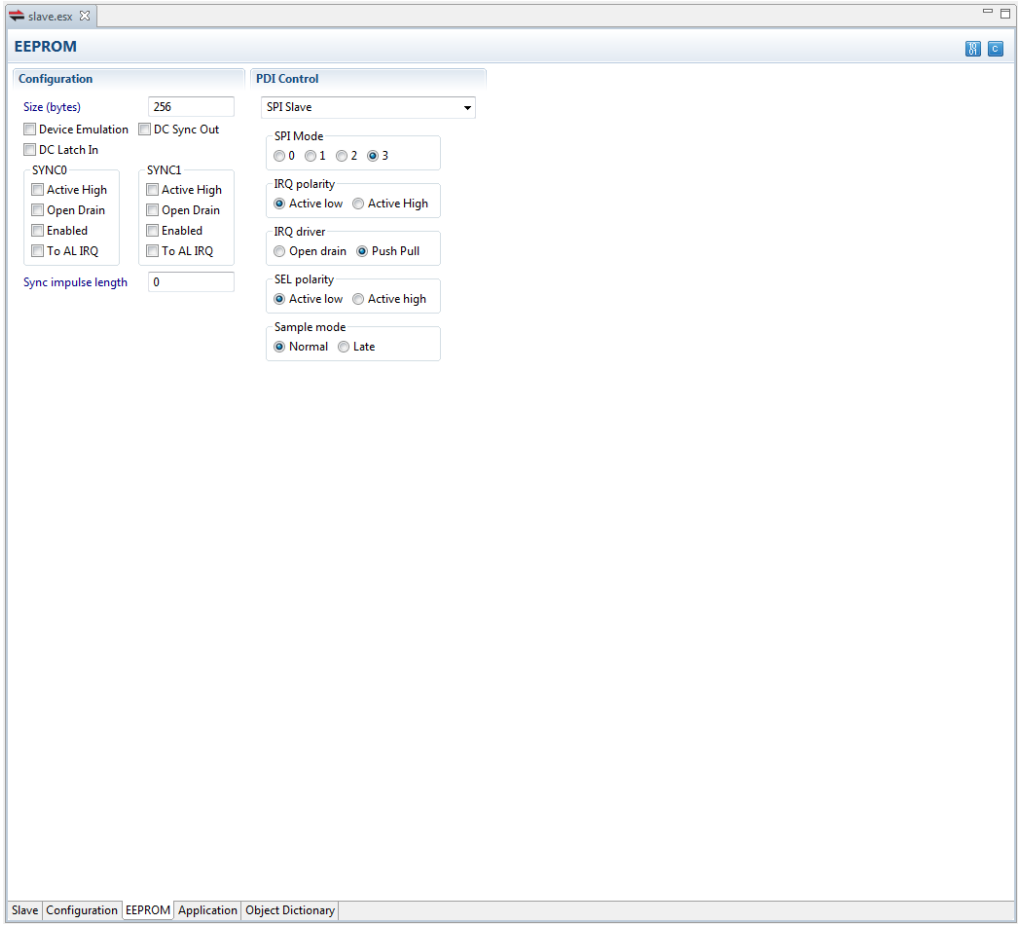
- Parameters for an advance slave with **CoE** and inputs and outputs are filled in by default



Slave configuration

EEPROM/ESC connection

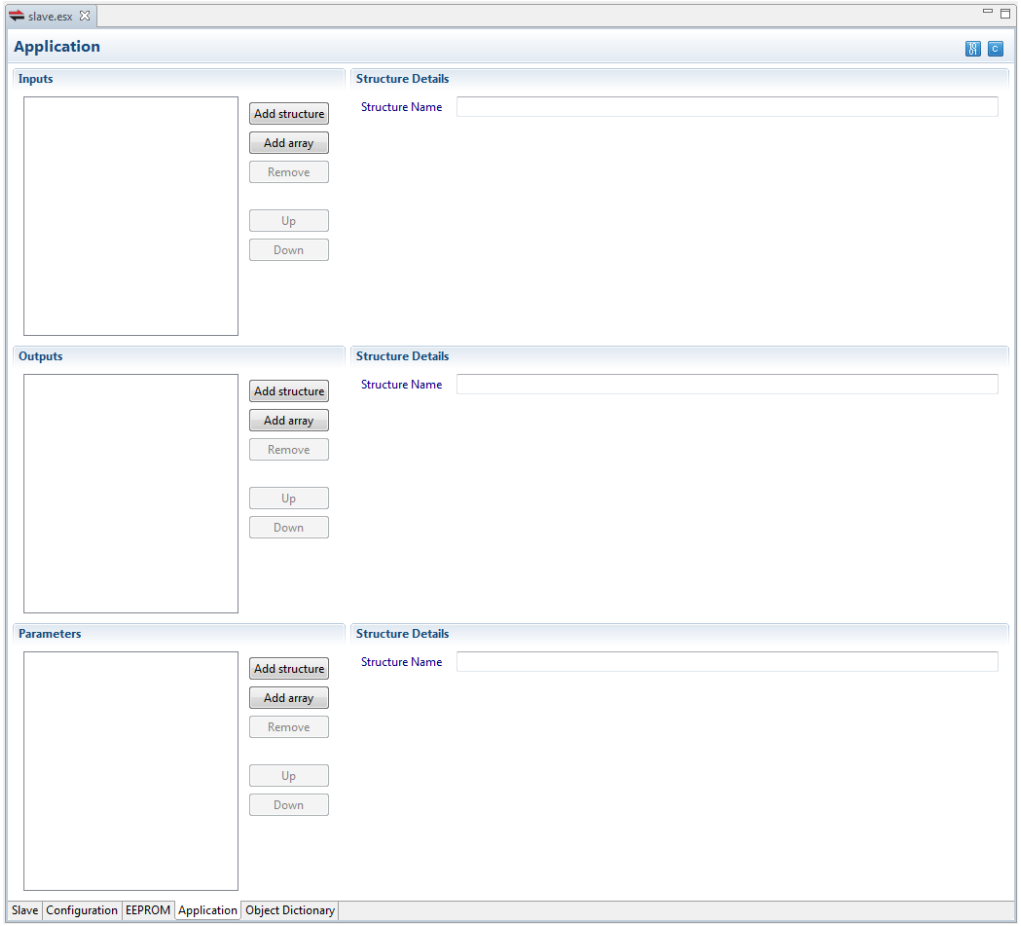
- This tab contains settings for the EEPROM such as the connection method of the ESC



Slave configuration

Process and service data

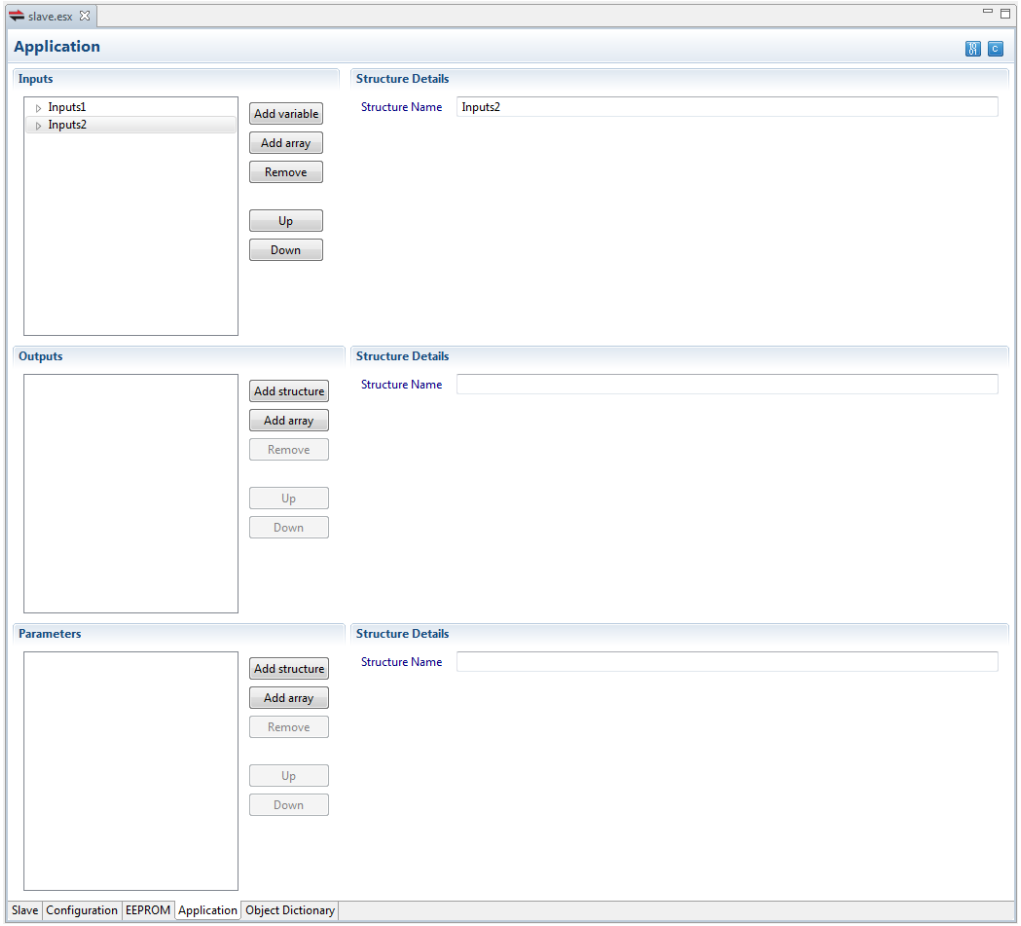
- In the Application tab we design the interface between the slave and the system
- In this tab we specify our PDOs and SDOs



Slave configuration

Process and service data

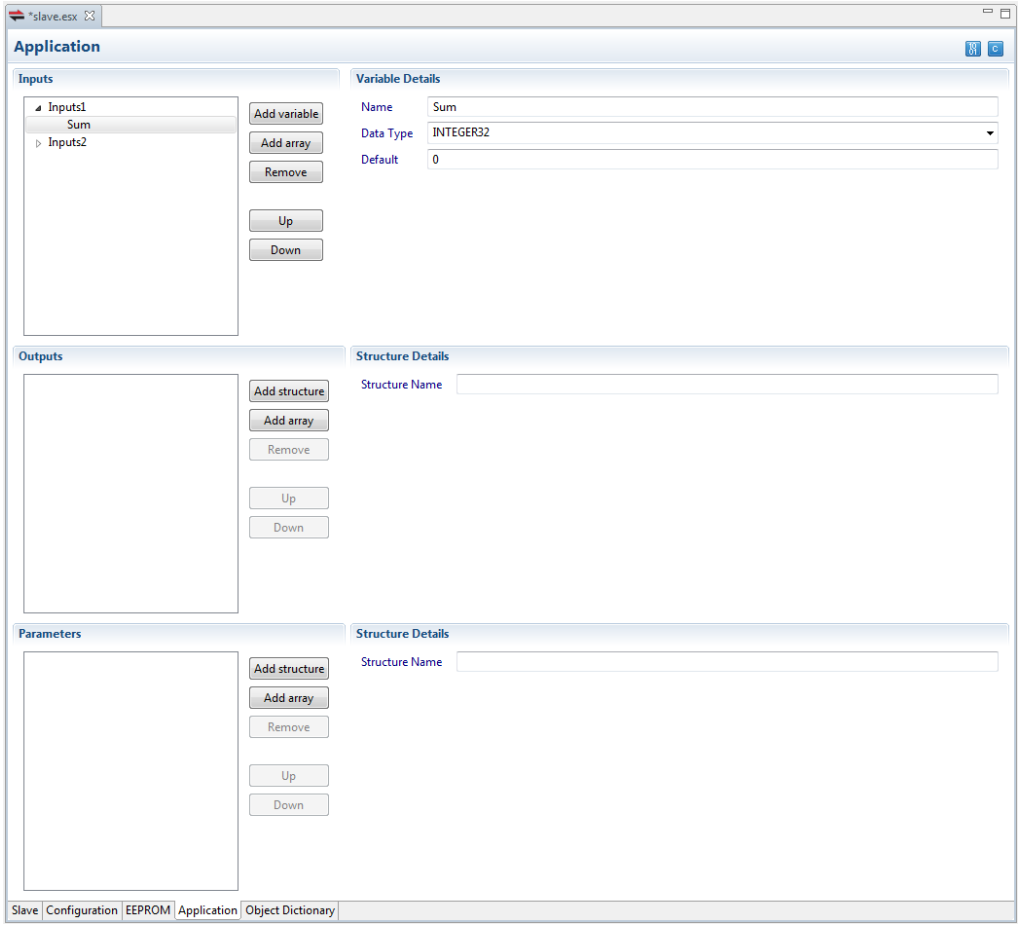
- We have the possibility to add multiple structures
- In this case we have created two structures which will later appear at different indexes in the object dictionary
- Inputs are the data that is sent from the slave



Slave configuration

Process and service data

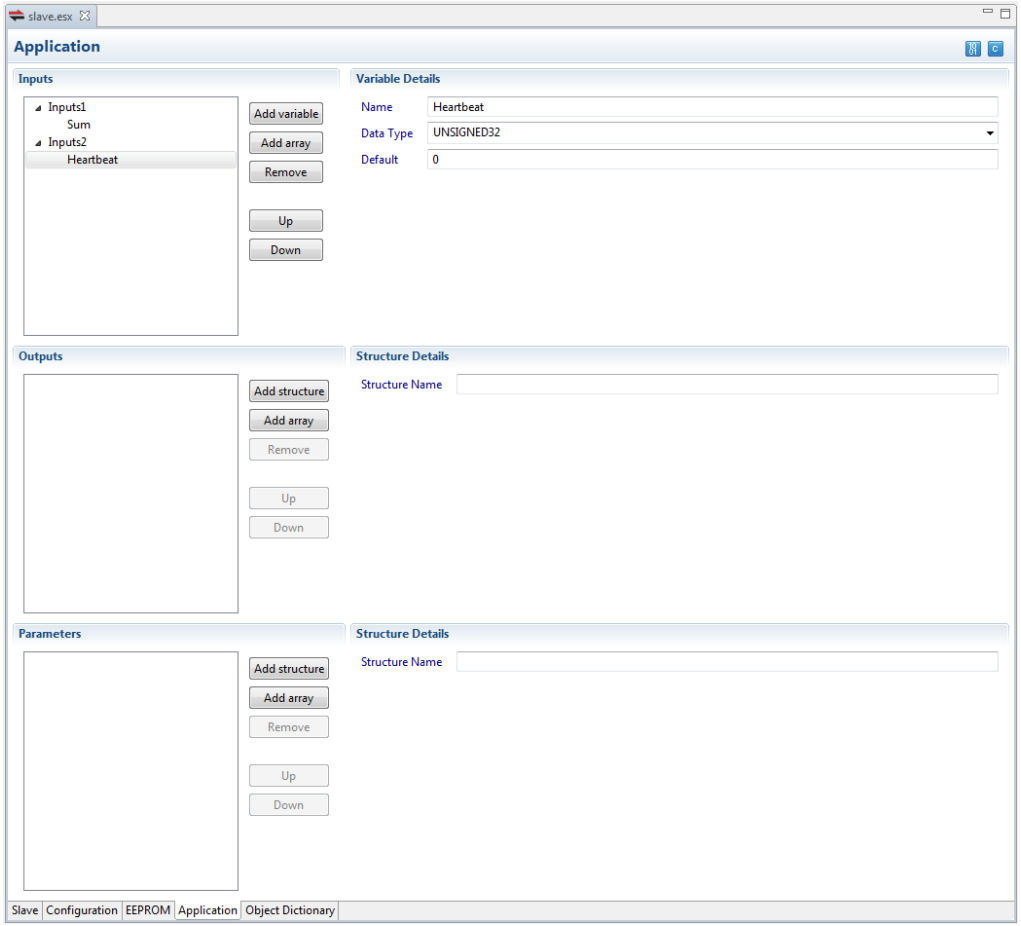
- Add a variable named **`Sum`** to the **`Inputs1`** structure



Slave configuration

Process and service data

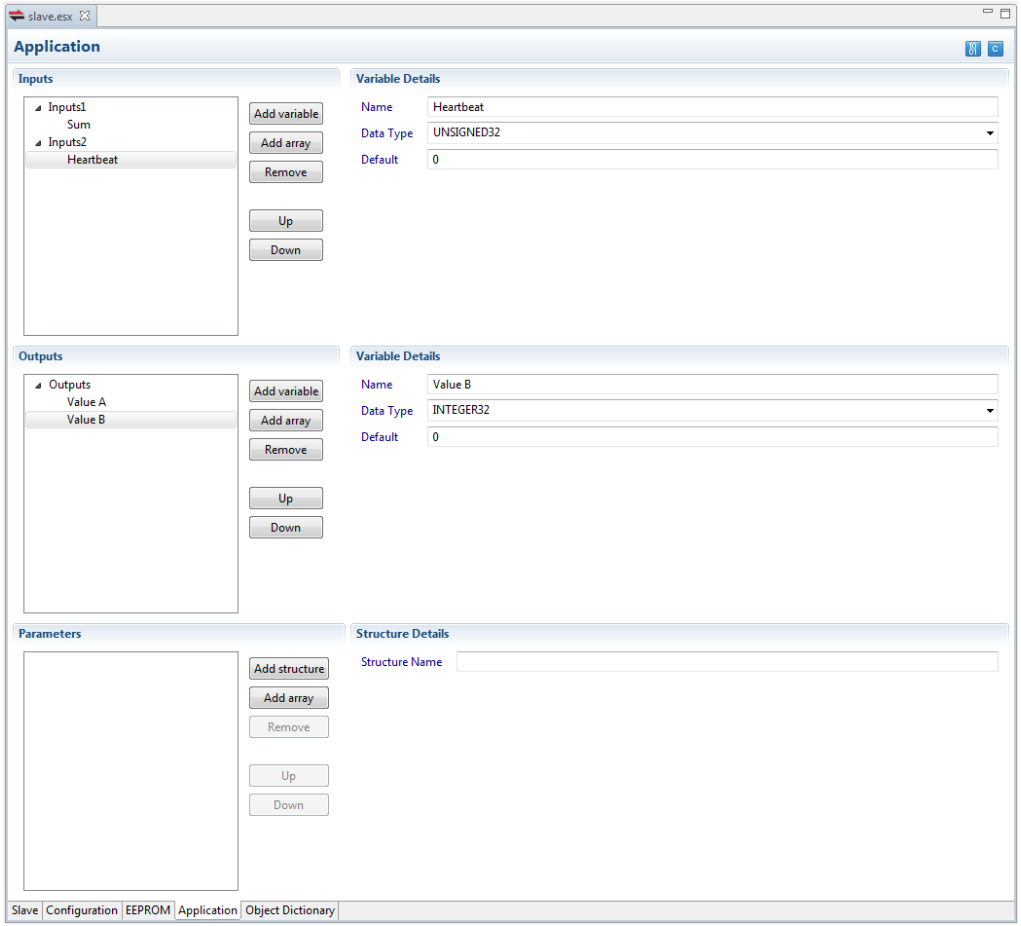
- Add a variable named **`Heartbeat`** to the **`Inputs2`** structure



Slave configuration

Process and service data

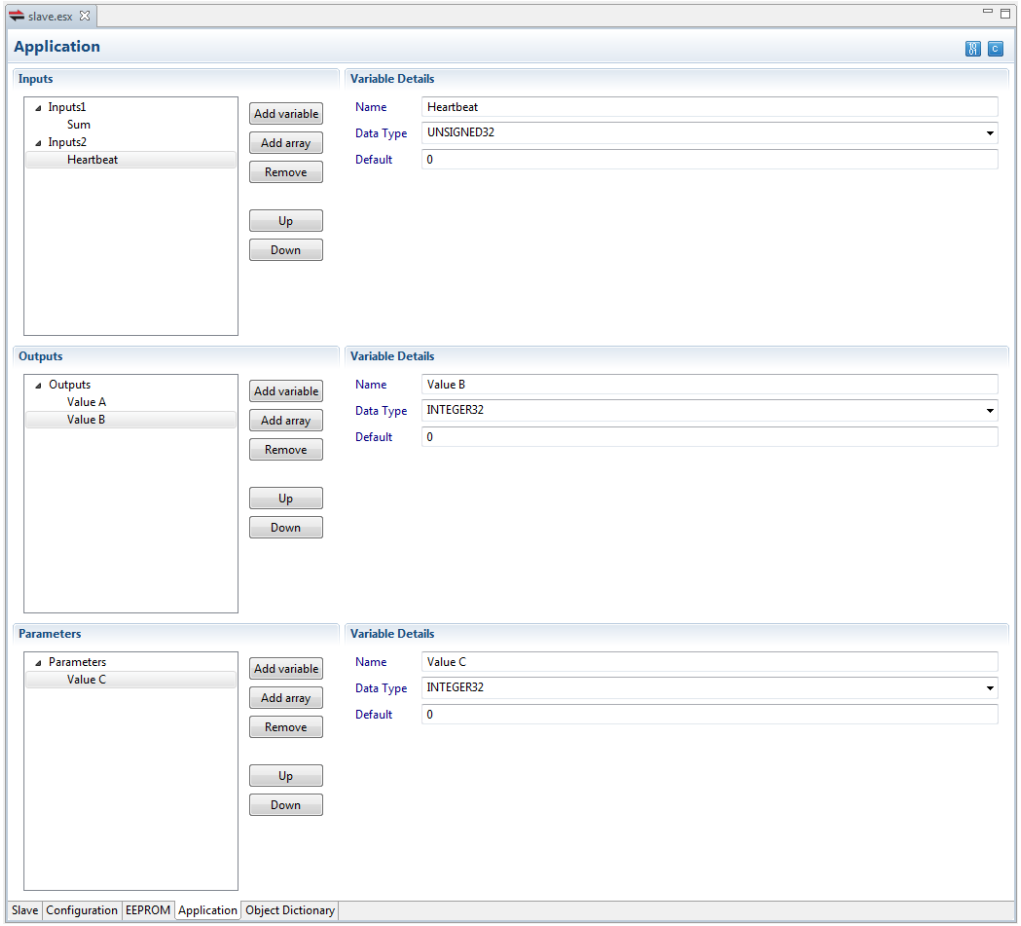
- Add an `Output` structure
- Add two `PDOs` to the `Output` structure



Slave configuration

Process and service data

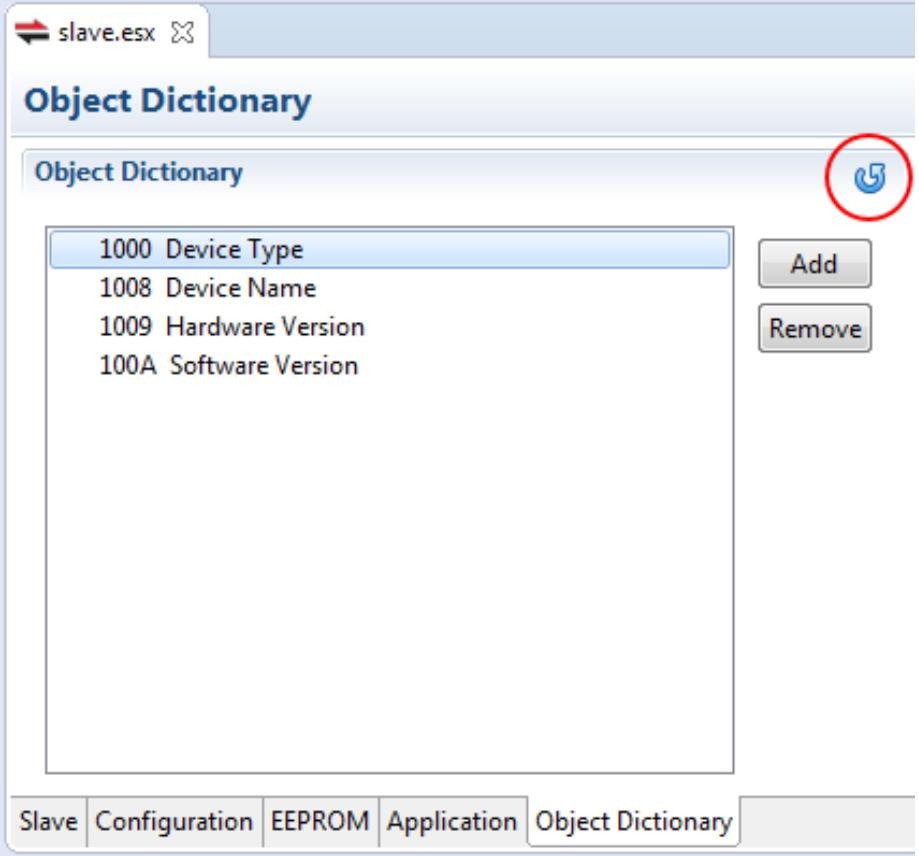
- And add an SDO as well



Slave configuration

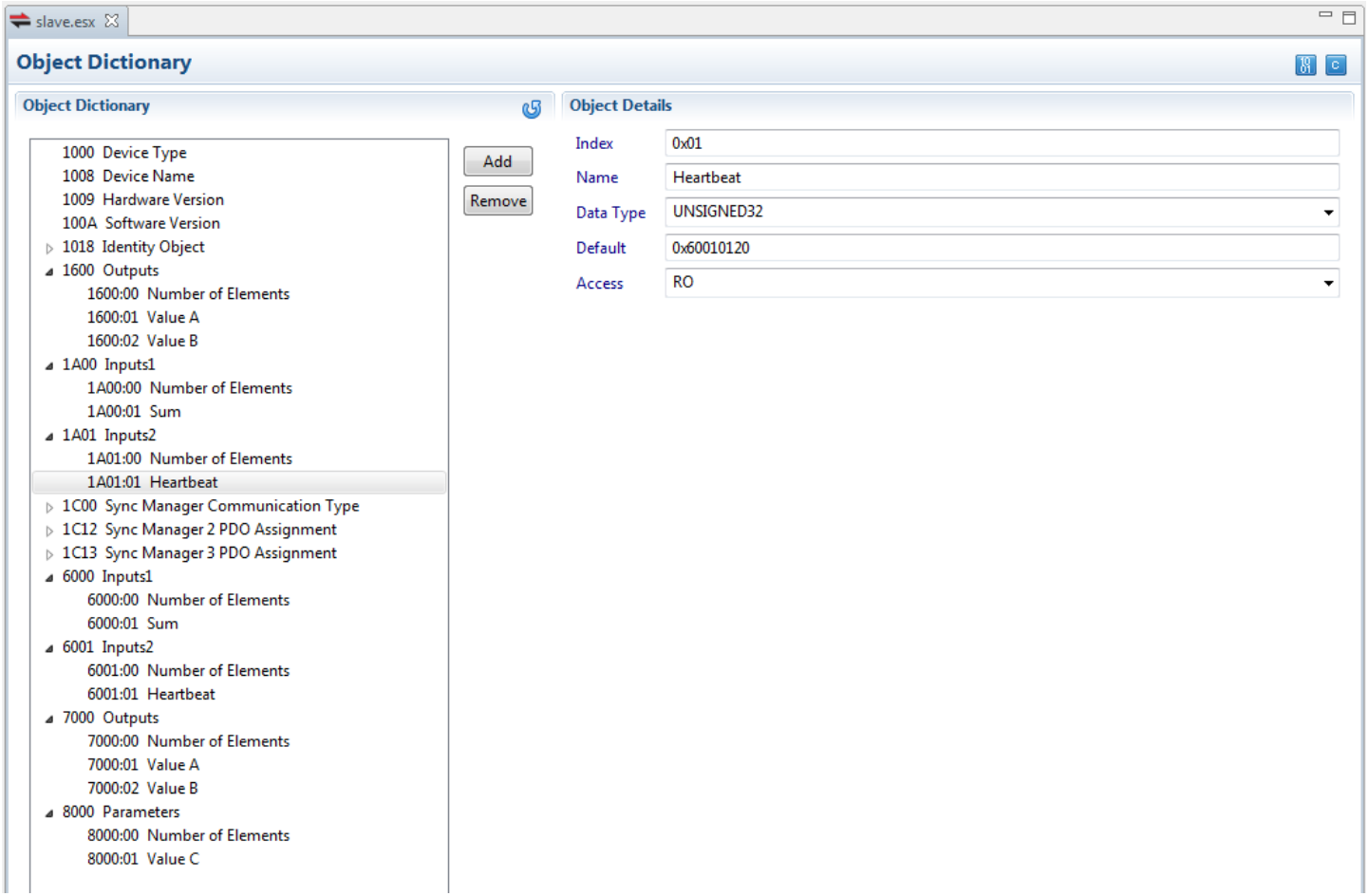
Object dictionary

- The Object dictionary tab provides a visualisation of the object dictionary
- To populate the object dictionary with the data supplied on the other tabs we need to synchronize it



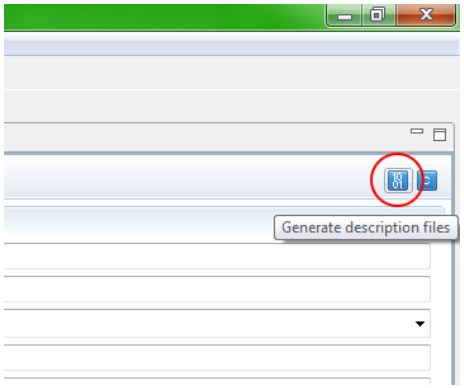
Slave configuration

Populated object dictionary

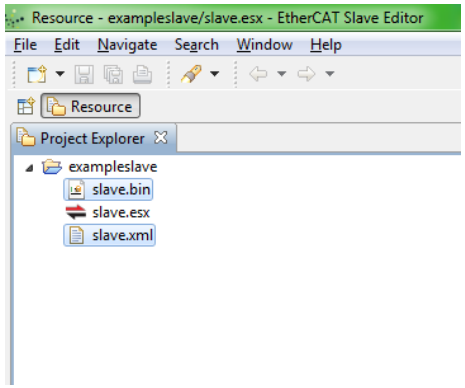


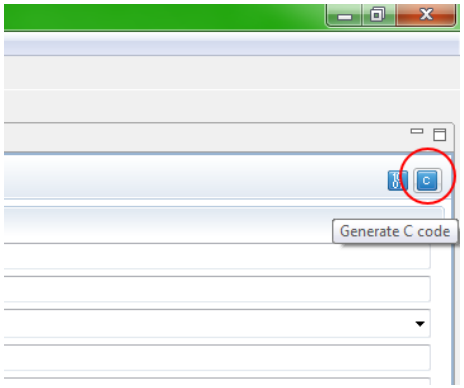
Generate files

EEPROM, ESI, code

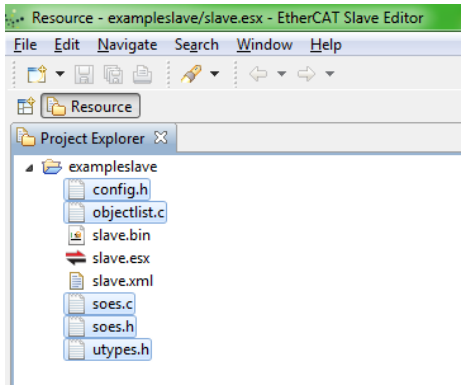


→





→



Download SOES

- The generated files could be used with `SOES`
- `SOES` is available for download from Open EtherCAT Society's GitHub page
- `https://github.com/OpenEtherCATsociety/SOES`

Implement callback functions

- All function interfaces are specified in **`soes.h`**
- We need to implement the callback functions to handle the PDO and SDO exchanges

```
/**
 * This function reads physical input values and assigns the
 * corresponding members of Rb.Inputs1
 */
void cb_get_Inputs1();

/**
 * This function reads physical input values and assigns the
 * corresponding members of Rb.Inputs2
 */
void cb_get_Inputs2();

/**
 * This function writes physical output values from the
 * corresponding members of Wb.Outputs
 */
void cb_set_Outputs();

/**
 * This function is called after a SDO write of the object
 * Cb.Parameters.
 */
void cb_post_write_Parameters(int subindex);
```

Example implementation of the callback functions

```
void cb_get_Inputs1()
{
   Rb.Inputs1.Sum = Wb.Outputs.Value_A + Wb.Outputs.Value_B + Cb.Parameters.Value_C;
}


void cb_get_Inputs2()
{
   static int32_t tick = 0;
   Rb.Inputs2.Heartbeat = tick++;
}


void cb_set_Outputs()
{
}


void cb_post_write_Parameters(int subindex)
{
   if (subindex == 1)
      printf ("Value␣C␣written␣->␣%d\n", Cb.Parameters.Value_C);
}
```